Navy Enterprise Maintenance Automated Information System (NEMAIS)

Testing Strategy

APP 003

Version 1.2

May 7, 2001

Table of Contents

Appendices

List of Figures and Tables

I. Referenced Documents

Client Strategy-System Landscape; January 22, 2001	Navy Enterprise Maintenance Automated Information System (NEMAIS) SAP Client Strategy, Version 1 (part of APP 006)
IEEE Std 610.12-1990 – IEEE Standard Glossary of Software Engineering Terminology	IEEE Standards Collection Software Engineering 1997 Edition
NEMAIS PMP; November 22, 2000	Navy Enterprise Maintenance Automated Information System (NEMAIS) Project Management Plan
IT Server Environment for Development/Test Implementation	Navy Enterprise Maintenance Automated Information System Work Product ARC-013
IT Process and Procedures	Navy Enterprise Maintenance Automated Information System Work Product ARC-018
Requirements Matrix	Navy Enterprise Maintenance Automated Information System Work Product BUS-011
High Level Business Process Flow	Navy Enterprise Maintenance Automated Information System Work Product BUS-013a
Detailed Business Process Flow	Navy Enterprise Maintenance Automated Information System Work Product BUS-013b
NETS POPP 2	Navy Enterprise Team Ships Program Office Policy and Procedure 2, Program Terminology (Rev A)
NETS POPP 8	Navy Enterprise Team Ships Program Office Policy and Procedure 8, Team Core Hours
NETS POPP 37	Navy Enterprise Team Ships Program Office Policy and Procedure 37, DOORS Application
Project Proposal – Volume II	Volume II of accepted proposal to RFQ N00024-00-Q-5229.
Configuration Management Plan	Navy Enterprise Maintenance Automated Information System (NEMAIS) CDRL 7

Complete Detailed List Library	Navy Enterprise Maintenance Automated Information System (NEMAIS) Work Product ENG-022
Integration Test Plan	Navy Enterprise Maintenance Automated Information System (NEMAIS) Work Product APP-070
User Acceptance Test Plan	Navy Enterprise Maintenance Automated Information System (NEMAIS) APP-076

1.0 PURPOSE

The purpose of this document is to define the Testing Strategy and establish the testing protocol that will be used by the Navy Enterprise Team Ships (NETS) in implementing Phase A of the Navy Enterprise Maintenance Automated Information System (NEMAIS) Enterprise Resource Planning (ERP) SAP solution and related "bolt-on" applications as they enable the business processes of regional and enterprise maintenance for Navy vessels. This document provides the Testing Strategy to be used for ensuring that the NEMAIS ERP solution meets all of the requirements identified to support enabling the Phase A "to-be" business processes in a timely and efficient manner. Refer to section 8.1 for further discussion on requirements and business processes. Subsequent NEMAIS project phase testing strategies will be based on this initial Testing Strategy document.

The NETS attaches a very high level of importance to thorough and effective testing at all stages of the NEMAIS project; therefore, it is imperative that all NETS project members review, understand and implement the testing processes identified in this document. Failure to adhere to the testing discipline could result in potential defects remaining undetected until after the "go-live" phase resulting in possibly serious or catastrophic failure of the NEMAIS business processes.

2.0 DOCUMENT INTERFACES

This document is considered a 'living' document and will be updated as other work products and events that feed this document are completed.

This document is intended to receive input from the following work products or events:
- BUS-001 Scope Refinement and Process Selection
- BUS-090 High Level Implementation Strategy
- Installation of Automated Testing Tools
- Installation of Requirements Management Tool

This document feeds the following work products or events:
- APP-054 Future Baseline Package System
- APP-060 Final Configured System and Unit Test
- APP-063 Integration Test Scenarios
- APP-070 Integration Test Plan
- APP-076 User Acceptance Test Plan

3.0 SCOPE OF THE PROJECT

NEMAIS will be implemented as a six-phase project, comprised of Phases A through F. The NEMAIS solution enables ship maintenance business processes spanning the hierarchies and the regions of the Navy from shipboard (O-level) to shore-based activities (Intermediate (I) and Depot (D) levels. A high level description of the project scope can be found in the NEMAIS

Project Management Plan (PMP). The Scope and Approach (ENG 020) document will provide the detailed description of the scope of this project from a solution perspective.

The components of the solution have been chosen for their ability to implement Naval maintenance processes across the entire Navy enterprise. The various components of the solution are illustrated in Section 6.2 of the PMP in Figure 5.

The core of the NEMAIS solution will be enabled by SAP's R/3 product, version 4.6c, with the Industry Specific – Public Sector (IS-PS)-2 industry-specific solution for US Federal accounting. All modules are available, but the initial design proposed centers the solution in Plant Maintenance (PM), with the other modules described in the PMP, Section 6.0, Table 8.

Refer to the PMP for additional detailed information regarding the NEMAIS solution.

4.0 TESTING GOALS AND OBJECTIVES

The goal of the Testing Strategy is to institutionalize a testing process that will verify the integrity of the system at specific stages of the project as illustrated in Figure 8.0-1. All business processes will be verified to work as designed and as expected by performing testing in a structured and disciplined manner. The ultimate goal of testing is to validate that the business can operate with sufficiently low risk on the new solution.

An objective for testing is, to the maximum extent possible, automatically validate all SAP configurations, ABAP development and bolt-on development before they are transported from development to test and on to the production environment, subsequently releasing the production version with no critical defects. The automated testing philosophy employed for this project consists of the following:

- automate tests wherever practical
- generate reusable scripts that can be used to test all modifications before production
- test all solutions selected for this implementation

The final milestone of testing is to conduct a successful User Acceptance Test (UAT) effort that results in an acceptance signoff by the nominated Navy representatives for the Phase A final system that meets or exceeds expectations.

The NEMAIS Testing Strategy is driven by the guiding principles of

- testing at the specified stages (see Figure 8.0-1)
- identifying defects early

The Testing Strategy process flow (Figure 8.0-1) illustrates the steps required at each testing stage to provide the building blocks necessary to complete successful Integration, Regression, System/Stress and User Acceptance Testing. Satisfactory execution of this process flow relies heavily on the collaboration of the Testing Team, Module Teams, Programming Team, Infrastructure Team, Basis Team and Change Management Team, as the project evolves and as virtual testing teams are built.

5.0 AUTOMATED TOOLS

Mercury Interactive has been selected as the automated testing tool vendor for the NEMAIS project. Four (4) of Mercury's test tools will be utilized. The test tools are TestDirector, WinRunner QuickTest for R/3, WinRunner and LoadRunner. The following sections describe each tool in more detail.

Telelogic has been selected as the software vendor to provide the automated requirements management tool for the NEMAIS project. Project requirements will be managed using Telelogic's Doors. Telelogic's DOORS and Mercury Interactive's TestDirector will be interfaced through Telelogic's DOORSConnect to manage and track the link between tests and requirements. The use of the Telelogic tools will be discussed in greater detail in the Configuration Management Plan and POPP 37 (DOORS Application).

The use of the automated tools selected for this project supports the automated testing philosophy by:

- supporting regression testing as defined in section 8.7
- allowing the Testing Team to develop automated testing scenarios for regression testing of business processes,
- allowing the Testing Team to run automated tests unattended 24 hours a day, if required
- increasing the speed of testing while maintaining the requisite accuracy
- reducing testing cycle time and resources required to execute tests
- simulating multiple persons performing entry of data
- providing reusable scripts for each stage of testing

5.1 TestDirector

TestDirector will be the repository of all tests and test sets that will be used to test the NEMAIS solution.

Tests will be recorded by designated Module team members using either WinRunner QuickTest for R/3 (section 5.2) or WinRunner (section 5.3) and saved in TestDirector. Tests may also be manually created and will be entered directly into TestDirector by designated Module Team members. The Test Leads will manage all tests stored in the TestDirector repository. Tests used for System/Stress Testing performed in LoadRunner will be selected by the Test Leads from the tests saved in TestDirector.

Test defects will be managed and reported from TestDirector. Refer to section 9.0 for additional information regarding defect management and tracking.

Testing metrics will be generated from TestDirector. Reports and graphs can be customized within TestDirector, as required, to provide information of interest at all levels within the project team. Refer to section 11.0 for additional information regarding testing metrics.

Tests will be linked to requirements through DOORSConnect, the interface between TestDirector and Telelogic's DOORS requirements management tool.

TestDirector provides the ability to automatically contact designated users via a rules based e-mail notification feature. Defect attribute fields can be selected to notify users according to their area of responsibility. The rules for automatic notification can be modified at any time during the project, and will be managed and monitored by the Test Leads to effectively use this feature without overburdening project resources with information overflow. TestDirector also provides the ability to manually e-mail defect information to selected users.

5.2 WinRunner QuickTest for R/3

WinRunner QuickTest for R/3 was developed with an understanding of the SAP R/3 graphical user interface (GUI) for the purpose of testing SAP R/3 applications.

Tests will be recorded in the SAP environment by designated Module team members using WinRunner QuickTest for R/3 and saved in TestDirector where design steps will be automatically created as part of the 'save' process. Situations may occur where automatic test script creation is not possible due to the nature of the configuration. In these situations, which are expected to be minimal, manual test script creation will be required. Manual scripts will also be saved in TestDirector by designated Module team members.

5.3 WinRunner

WinRunner will be used to record tests for all non-SAP applications. Tests recorded using WinRunner will be saved in TestDirector by designated Module team members. Situations may occur where automatic test script creation is not possible. In these situations, which are expected to be minimal, manual test script creation will be required. Manual scripts will also be saved in TestDirector by designated Module team members.

5.4 LoadRunner

LoadRunner will be used for System/Stress Testing of the NEMAIS "to-be" solution. LoadRunner licenses for 2000 virtual users have been procured for Phase A of this project and will be used to support simultaneous testing of the system. The number of LoadRunner licenses procured for Phase A is based on testing the system to twice the number of users on line at anyone time. For the Norfolk SIMA there are about 2500 users of which approximately one-third are online at any one time. Future NEMAIS phases will require additional LoadRunner licenses proportional to the estimated number of users for each phase implementation.

Tests will be selected from the automated tests saved in TestDirector for use with the LoadRunner testing tool. Details of the selection process will be included in the System Test Plan.

6.0 CLIENT STRATEGY

The Client Strategy-System Landscape (part of APP 006) defines the five instances that will be used during the NEMAIS project. The document also defines the transport strategy and clients included within each instance. It is essential that the reader become familiar with the contents of the Client Strategy-System Landscape to assist in better understanding the remaining content of the Testing Strategy.

The following are the assignments for the five instances defined in the Client Strategy-System Landscape:

1. DEV (used for Development)
2. TST (used for Testing)
3. PRD (used for Production)
4. TRN (used for Training)
5. SHO (portable, used for Road Show Demos)

Transport Management is the SAP tool used for moving configuration and object changes from one instance to another. The approval workflow for transports is defined in the Client Strategy-System Landscape and will govern the promote to production process for the NEMAIS project.

7.0 INFRASTRUCTURE TESTING

Infrastructure Testing is performed by the Infrastructure and Basis Teams and is comprised of testing for Failover, Hardware/Software Validation, Disaster Recovery, Failure Component and Backup and Recovery. Definitions for each of these types of testing are documented in Appendix B. The Infrastructure Team will document the details of Infrastructure Testing in the NEMAIS project work products ARC-013 - IT Server Environment for Development/Test Implementation and ARC-018 - IT Process and Procedures. Refer to these documents for detailed information regarding Infrastructure Testing.

8.0 TESTING STRATEGY PROCESS FLOW

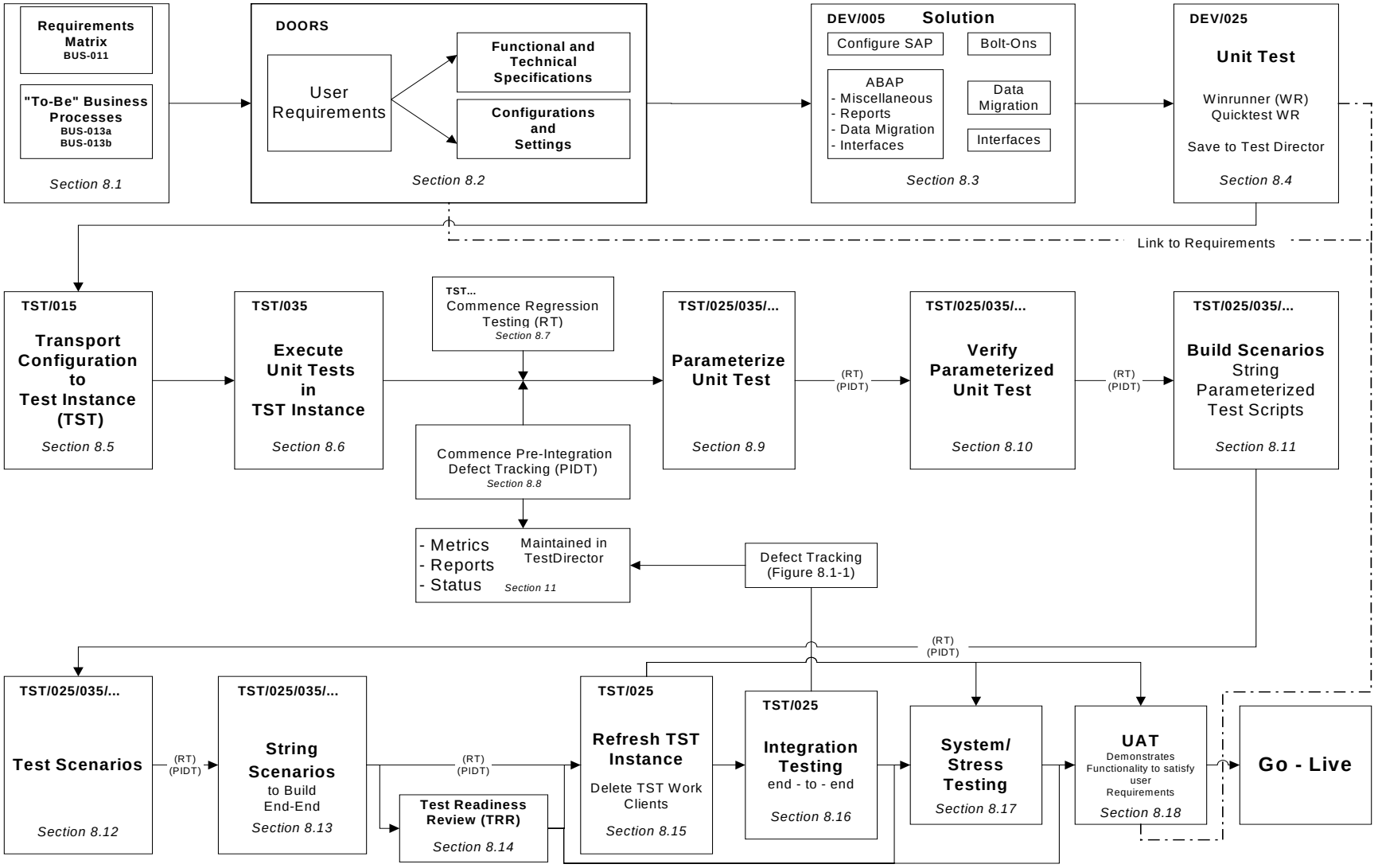
Figure 8.0-1 illustrates the process workflow associated with the NEMAIS project Testing Strategy. The following subsections provide details on the actions performed within each workflow step. Each workflow box includes the identification of the associated Test Strategy subsection and the TST client supporting the performance of each process flow step.

Satisfactory completion of each process flow step requires the involvement of Module Team members knowledgeable in the affected business areas and the associated functionality requirements to ensure the NEMAIS testing objectives and test schedules are met.

Satisfactory completion of each process flow step includes the verification of associated tests, test sets, and test scenarios at key points in the process flow illustrated in Figure 8.0-1. These key points are discussed in the following sub-sections.

Verification ensures that test scripts operate as intended, that scripting errors are not present, and that all required input data is valid. Verification of an automated test will require successful playback of the test script in its entirety with the actual test results matching the expected designed results. Verification of a manual test requires that actual test results match the expected designed results.

Figure 8.0-1 Testing Strategy Process Flow



8.1 Business Processes and Requirements

Business processes for the "to-be" environment will be identified and documented in two separate work documents, BUS 013a for the High Level Business Process Flow and BUS 013b for the Detailed Business Process Flow. The business process requirements will be documented in BUS 011 Requirements Matrix. The information in each of these documents will be entered and maintained in the requirements management tool Telelogic's DOORS.

8.2 Configurations and Settings/Functional and Technical Specifications

The configuration process will begin as user requirements are identified and determined to be in scope for Phase A of the NEMAIS project. The Module Teams will determine configurations and settings and will identify gaps in functionality and any ABAP development necessary to satisfy the user requirements. Functional and technical specifications will be generated by the Data Migration and Programming Teams and other teams responsible for interfacing the bolt-ons with the SAP R/3 solution.

Identifiers for configurations and settings and functional and technical specifications will be entered in the requirements management tool, Telelogic's DOORS, and linked to user requirements and business processes as discussed in POPP 37 DOORS Application.

8.3 Solution

Module Teams will configure the SAP R/3 system and generate requests for ABAP development to satisfy requirements of the NEMAIS solution. ABAP development may be required for functionality gaps, customized reporting, data migration, interfaces and bolt-ons. As configurations are made and ABAP code developed, Unit Configuration Testing and Unit Testing of Development will be performed, respectively. These tests are described in more detail in the next section.

8.4 Unit Test

Unit tests will be created in the DEV instance by the Module Teams and Programming Team using the Mercury Interactive test suite. In contrast to unit testing performed during standard software development projects, SAP R/3 projects perform two different types of unit testing, Unit Configuration Testing, performed by the Module Teams, and Unit Testing of Development, performed by the Programming Team. Unit testing is often referred to as "White Box" testing where the internal logic of the software program is tested. For SAP R/3, which is a commercial off the shelf product, no "White Box" testing will be performed since the SAP R/3 system is essentially a "Black Box" to the individuals configuring the software. Therefore, Unit Configuration Testing can be referred to as "Black Box" testing, where the input and expected results are devised based only on the requirements with no need for knowledge of the software's internal logic. Unit Testing of Development, which is most similar to standard software development unit testing where the internal logic is of concern, can be referred to as "White Box" testing.

Naming conventions for unit tests will be described in a later document or added as an Appendix to this document after the automated testing tool and requirements management tool have been

installed and configured. All unit tests recorded with WinRunner QuickTest for R/3 and WinRunner will be saved in Mercury Interactive's TestDirector and any test that cannot be automated will be manually created and loaded into TestDirector. The saved tests will include the information required to link the tests to the corresponding detailed process or processes documented in Telelogic's DOORS as BUS-013b. The specific information required to create these links will be captured in TestDirector user defined fields created after BUS-013b is completed.

As discussed in section 5.0, tests will be linked to requirements using Telelogic's DOORSConnect. General (BUS 013a) and detailed (BUS 013b) business processes and associated requirements (BUS 011) will be maintained in Telelogic's DOORS as separate modules. In addition, a module created to capture the test information transferred through Telelogic's DOORSConnect from TestDirector will be used to link tests at the unit level prior to the transport of the most recent configuration from DEV 005 to the TST instance.

Prior to recording a unit test both the data and the screen flow associated with a configured transaction must be determined to ensure the test is recorded efficiently and accurately. WinRunner QuickTest for R/3 and WinRunner capture the data used during the recording of a test as a constant value. As a result, this data is used each time the test is executed. Therefore, it is imperative that the configuration tested by a unit test be checked to ensure that the reuse of the same data for each additional test execution will not cause failure. If the functionality of the configuration requires unique data for each unit test execution, the recorded test script must be modified in order to support the completion of unit testing in the DEV instance. This modification is referred to as parameterization. Initial parameterization will be performed by the Module Teams. The Testing Team will perform additional parameterization as addressed in section 8.9. Regardless of the type of data required, each unit test will be verified as described in section 8.0 prior to a transport occurring. Completion of this verification will be captured by the promote to production process (part of APP 006) which governs the documentation for validating transports.

Automated and manual unit tests, for a given transport, will be verified by the Module Teams prior to performing the transport to the TST instance and other instances.

There may be situations in which configurations for a given transport are dependent on configurations to be included with future transports. Unit tests associated with these configurations will be categorized as "on-hold" and will be tracked and managed in Test Director. In addition, the number of tests in this "on-hold" category will be reported as a key metric during the configuration phase.

8.4.1 SAP Unit tests using WinRunner QuickTest for R/3

Module Teams configuring SAP will record unit tests using Mercury Interactive's WinRunner QuickTest for R/3. The recorded tests will be saved in Mercury Interactive's TestDirector. The act of recording the test will create a QuickTest script that can be subsequently executed as required. TestDirector will automatically create test steps for each recorded test as part of the save process.

Each individual that creates a unit test should review the test steps created by TestDirector and modify the steps as required for completeness and accuracy. As previously discussed, the data used to execute the test is captured when the unit test is recorded. Any additional information or special instructions regarding the data should also be added to the test steps. User fields will be created in TestDirector to capture the transport number of the configuration covered by a given unit test.

It is considered best practice and both preferred and recommended that unit test scripts be recorded and verified by the Module Teams before the configuration is keyed into the Master Client 005 of the DEV instance. After the configuration has been entered in Master Client 005 and transported back to clients in the DEV instance, as described in the Client Strategy-System Landscape, each test should again be verified by the Module Teams. It is important to perform this second verification to ensure that the test script operates as intended and that tests do not fail due to errors in scripts, data or errors associated with the manual entry of the configuration into DEV 005. Unit tests are subject to configuration management once the transport occurs.

8.4.2 Unit tests using WinRunner

For any non-SAP testing, unit tests will be recorded using the WinRunner tool. Recorded tests and any manually created tests will be saved in TestDirector. It is important that Module Teams ensure that unit test scripts operate as intended and that tests do not fail due to errors in scripts or data.

8.5 Transport to TST Instance

New configuration will be transported to the TST instance from the DEV instance as required. Transports are governed by the contents of the Client Strategy-System Landscape document.

8.6 Execute Unit Tests in TST instance

Once a transport has been imported into the TST instance, new and modified unit tests for that transport will be executed by the Testing Team using the TST instance. Execution of the tests in TST ensures the integrity of the configured solution associated with each test was maintained through the transport process.

8.7 Regression Testing

After each transport to the TST instance, regression testing will be performed by the Testing Team using the TST instance to verify the integrity of the transport. The Test Leads will create a regression test set in TestDirector that will contain all tests to be included in the regression test run. Unit tests associated with any given transport will be included in the regression test set after the tests have been verified by the Testing Team in the TST instance, as discussed in section 8.6. Before executing the regression test run, the Test Leads will review the regression test set and will exclude any unit tests associated with modified configuration for the current transport, since these tests require re-execution in the TST instance (section 8.6) prior to inclusion in the regression test set. Unit tests identified as ”on-hold”, as discussed in Section 8.4, will also be excluded from the regression test set.

8.8 Commence Pre-Integration Defect Tracking

Any defects encountered after executing the unit tests in the TST Instance, section 8.6, and up to the commencement of Integration Testing, section 8.16, will be managed within TestDirector as pre-integration test defects. Communication with the Module Teams will be critical to ensure quick removal of these defects so there is no negative impact on either new configuration efforts or configurations dependent on resolution of defects.

8.9 Parameterize Unit tests

Once a unit test is verified in the TST instance, the test will be modified by the Testing Team to use data sets that can be fed from Excel spreadsheets. These data spreadsheets can be generated manually or from databases. Parameterized tests will be renamed using the convention <testname-p> to indicate parameterization. Additionally, a user field in TestDirector will be created to track whether a test is parameterized and will be set to "Y".

Detailed information regarding the naming of tests throughout the testing process for all tests will be included in a later document or added as an Appendix to this document after the automated testing tool and requirements management tool have been installed and configured.

8.10 Verify Parameterized Unit Tests

The Testing Team will be responsible for verifying the parameterization modifications made to unit tests to ensure that the scripts are performing correctly and that any data provided is appropriately cleansed and accurate. After verification, unit tests can be combined in test sets to correspond to business scenarios.

8.11 Build Scenarios (Using Parameterized Unit Test Scripts)

Within TestDirector, tests will be grouped in test sets that will be used to verify the operation of a business scenario. Unit tests will be strung together in test sets to create realistic and integrated business process scenarios based on the information provided by the Module Teams in BUS 13b Detailed Business Process Flow. Only unit tests that have been parameterized and verified will be included in these test sets.

8.12 Test Scenarios

Once test sets are built they will be verified by the Testing Team to ensure that the test flow is correct and to ensure that scripts still operate with data provided. Any scripting or stringing defects encountered will be tracked, resolved, re-tested and cleared in a timely fashion.

8.13 String Scenarios

Verified test scenarios described in Section 8.12 will be combined in larger test sets by the Testing Team, with input from the Module and Business Transformation Teams, to begin building the end to end testing scenario required during integration testing. These end-to-end testing scenarios will be verified and corrected as necessary.

8.14 Test Readiness Review

A Test Readiness Review (TRR) will be conducted no less than two (2) weeks prior to the scheduled start of Integration Testing. The details of the test readiness review will be provided in the Integration Test Plan. A TRR will also be performed prior to System Testing and UAT.

8.15 Refresh TST Instance

As described in the Client Strategy-System Landscape (part of APP 006), client 025 in the TST instance will be refreshed to prepare for Integration Testing. This refresh will ensure TST 025 is prepared for performing the necessary data loads, end-to-end testing and testing of user profiles during the integration test effort. The TST instance will also be refreshed prior to System Testing and UAT. Refer to the Client Strategy-System Landscape document for a detailed description of the refresh process.

8.16 Integration Testing

Integration Testing will be performed to validate that all of the software components of the NEMAIS solution work properly together and operate according to the requirements (BUS-011 – Requirements Matrix) generated for this project.

As depicted in the Testing Strategy Process Flow (Figure 8.0-1), Integration Testing will be an evolutionary process that is driven by and builds upon previous testing efforts. As discussed in section 8.11 the test scenarios that were developed from unit testing will be reviewed and, where appropriate, selected and evolved into a realistic and integrated business process scenario or process flow. Where necessary, more than one scenario for a given business process may be evolved to ensure that all significant process variants are included for testing.

Integration Testing will focus on cross-functional integration points, including external interfaces, as well as end-to-end business processes. Since knowledge of the internal logic of the software is not required to support integration testing, it is often referred to as “Black Box” testing. A well defined Integration Test Plan (APP 070) will be prepared by the Testing Team, with input from the Module Teams, that schedules resources, including testers, and tests to be performed to satisfy the Integration Test. The test plan will also address the exit criteria to be reviewed and approved by applicable stakeholders and NETS management.

Integration Testing is scheduled to begin approximately four (4) months before the go-live date, will be conducted for a period of six (6) weeks and will require a resource team of approximately twenty (20) to twenty-five (25) testers. This virtual testing team will be made up of members from the NETS team, including consultants and Navy personnel, subject matter experts and module experts. Testers on this virtual testing team must be 100% dedicated to the scheduled six (6) week Integration Test effort in order to complete the effort in the time allotted. It is important to have segregation of duties between testers and configurers in order to more efficiently resolve defects and to avoid delays in testing. While not rigidly enforced, this segregation can be achieved by leveraging the Business Transformation and Module Team distinctions. Module Team members will be responsible for resolving defects and Business Transformation Team members will be responsible for performing the Integration Testing. The Integrated Test Plan will expand upon the roles and responsibilities of the Testing Team.

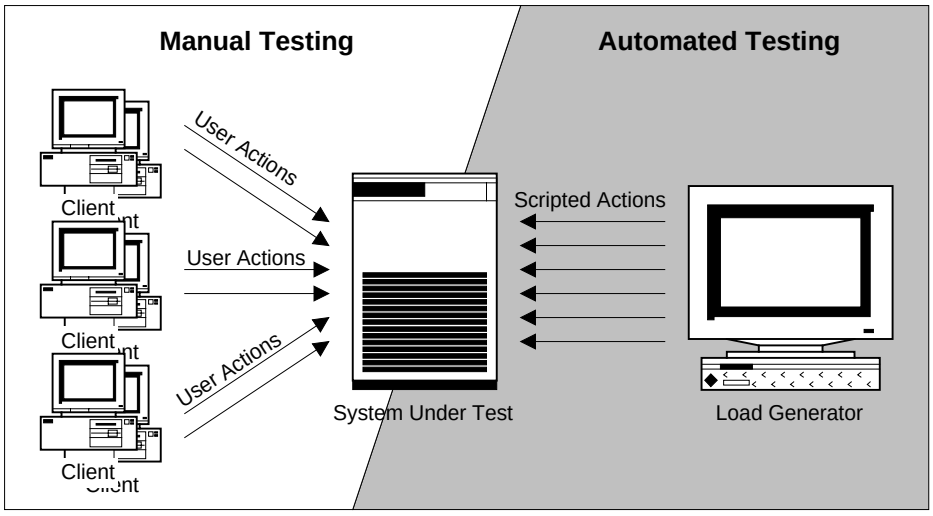
A facility must be ready two (2) weeks prior to the start of Integration Test to allow for setup and shakedown. The facility must have connectivity to both the testing tools and the TST instance. Additional details for facility requirements will be described in the Integration Test Plan.

Defects will be tracked and managed throughout the integration process using TestDirector. Refer to the section entitled Defect Tracking in this document for further discussion of defects.

A number of metrics, reports and status will be provided during the Integration Testing effort that will be available in TestDirector and can be made available on the NETS website. Refer to section 10.0 in this document for additional details regarding testing metrics for this project.

8.17 System/Stress Testing

System Testing will commence when Integration Testing is complete, will be performed in the same facilities used for Integration Testing, and is scheduled as a three (3) week event. Responsibility for the planning and conducting of System Testing will be shared by the IT Architecture Team, Basis Team and Testing Team. A well-defined plan will be created by these teams to govern the System Testing effort. This plan will become part of ENG-022 Complete Detailed Test Library. A TRR will be performed and the TST instance will be refreshed prior to commencing testing as illustrated in Figure 8.0-1. For the NEMAIS system, load and stress tests will be performed to validate both the general system acceptability and 'stress' test performance. Stress testing involves the execution of a pre-defined series of tests against the system under test (SUT). These tests are typically executed at high, concurrent volumes, making it impractical to assemble hundreds or thousands of computer systems, with operators, to generate the desired load. For this project Mercury Interactive's LoadRunner will be used. LoadRunner enables user actions to be scripted and later replayed upon demand. Once created, the script can be used to generate the workload of tens or hundreds of virtual users on a single, powerful computer as depicted in Figure 8.17-1.



Actions of multiple real clients (left) can be simulated by a single load generator (right)

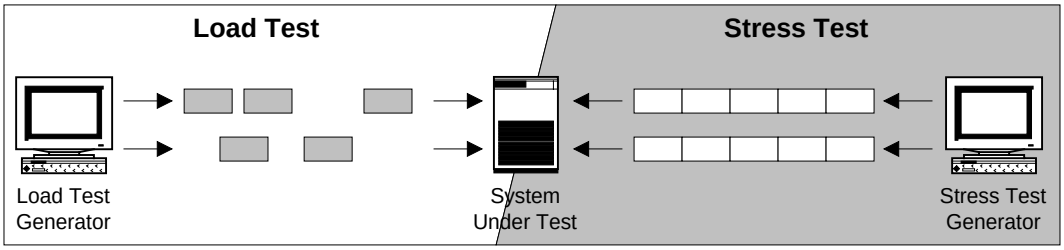
Figure 8.17-1 Manual vs Automated Testing

LoadRunner licenses for 2000 virtual users have been procured for this project and will be used to support simultaneous load testing of the system.

If volumes of traffic required are greater than one client system can generate, multiple client systems can be concatenated together and centrally controlled.

The objective of load testing for the NEMAIS project will be to determine how the SUT performs under an artificially generated user load based upon real world execution scenarios. These scenarios will include typical transaction mixes, transaction arrival rates, and user think time.

Stress testing is not intended to simulate real world usage. Instead, it is used to determine the breaking point of a specific subsystem or set of subsystems by placing an inordinate amount of stress upon the SUT. Unlike load testing, stress test cases executed against the SUT do not contain user think time. Instead, test cases are executed as quickly as possible in a back-to-back fashion. Stress tests may or may not contain a transaction mix, depending upon test objectives. Despite different objectives, both load and stress testing utilize the same tool set and test methodology. Figure 8.17-2 graphically depicts the differences between load and stress testing.



Load tests (left) include think time between virtual user actions; stress tests (right) do not.

Figure 8.17-2 Load vs Stress Test

8.17.1 Performance Criteria

Performance criteria and response time criteria are defined in the response to RFQ N00024-00-Q-5229. The criteria listed in Table 8.17.1-1 is consistent with industry standards and will be used for reference purposes only in this document and to demonstrate the types of criteria that will be used for the System Test.

Table 8.17.1-1 Performance Criteria from Proposal

Performance Test	Threshold
Time for initial screen display to appear	2 sec
Delay before “processing: Please Wait” message appears	2 sec
Simple form display: data from less than three dynamic tables and five lookup tables	1 sec
Complex form display: data from three or more dynamic tables and five or more lookup tables	2 sec
Simple form update: data from less than three dynamic tables and five lookup	1 sec

tables.	
Complex form update: data from three or more dynamic tables and five or more lookup tables	2 sec
Display 200,000 byte graphic image	3 sec
Simple query of 100,000 database records from a single table, select and display list of 500	10 sec
Complex query of 100,000 database records joining three or more tables, select and display list of 500 records	30 sec

8.18 User Acceptance Testing

User Acceptance Testing will commence when Systems Testing is complete. Responsibility for the planning and oversight of User Acceptance Testing will be shared between the Change Management Team and the Testing Team. A well-defined User Acceptance Test Plan will be created by the Testing Team, with support from the Change Management Team, that schedules resources and tests to be performed to satisfy the User Acceptance Test. The plan will also address the performance of a TRR and a refresh of the TST instance prior to commencing testing as illustrated in Figure 8.0-1. Additional responsibilities are discussed in section 13.0, Roles and Responsibilities.

User Acceptance Testing is scheduled as a three (3) week event and will be performed in the same facility used for the Integration Testing effort. User Acceptance Testing will be performed by a group of users that have been identified and scheduled by the Change Management Team. Selection of these users should be completed at least 3 months before the commencement of UAT to allow for the completion of administrative and training requirements. Personnel assigned to the user group should be temporarily assigned to the NETS team to ensure they are capable of providing 100% dedication to the testing effort. The group should be made up of twenty (20) to twenty-five (25) members from the user community.
Users selected for this effort should:

- understand the core business processes that are included in the NEMAIS solution
- represent a cross section of relevant functional experience with respect to the core processes included in the NEMAIS solution.
- be recognized within the user community as representatives and spokespersons for the user community

Once selected these users should receive, prior to the commencement of UAT

- applicable user training on the SAP application solution, including any associated bolt-ons, for the role they will perform
- training on the “To Be” processes included in the NEMAIS solution
- training on the Mercury Interactive Tool Set
- training on Requirements Traceability, including Telelogic’s DOORS

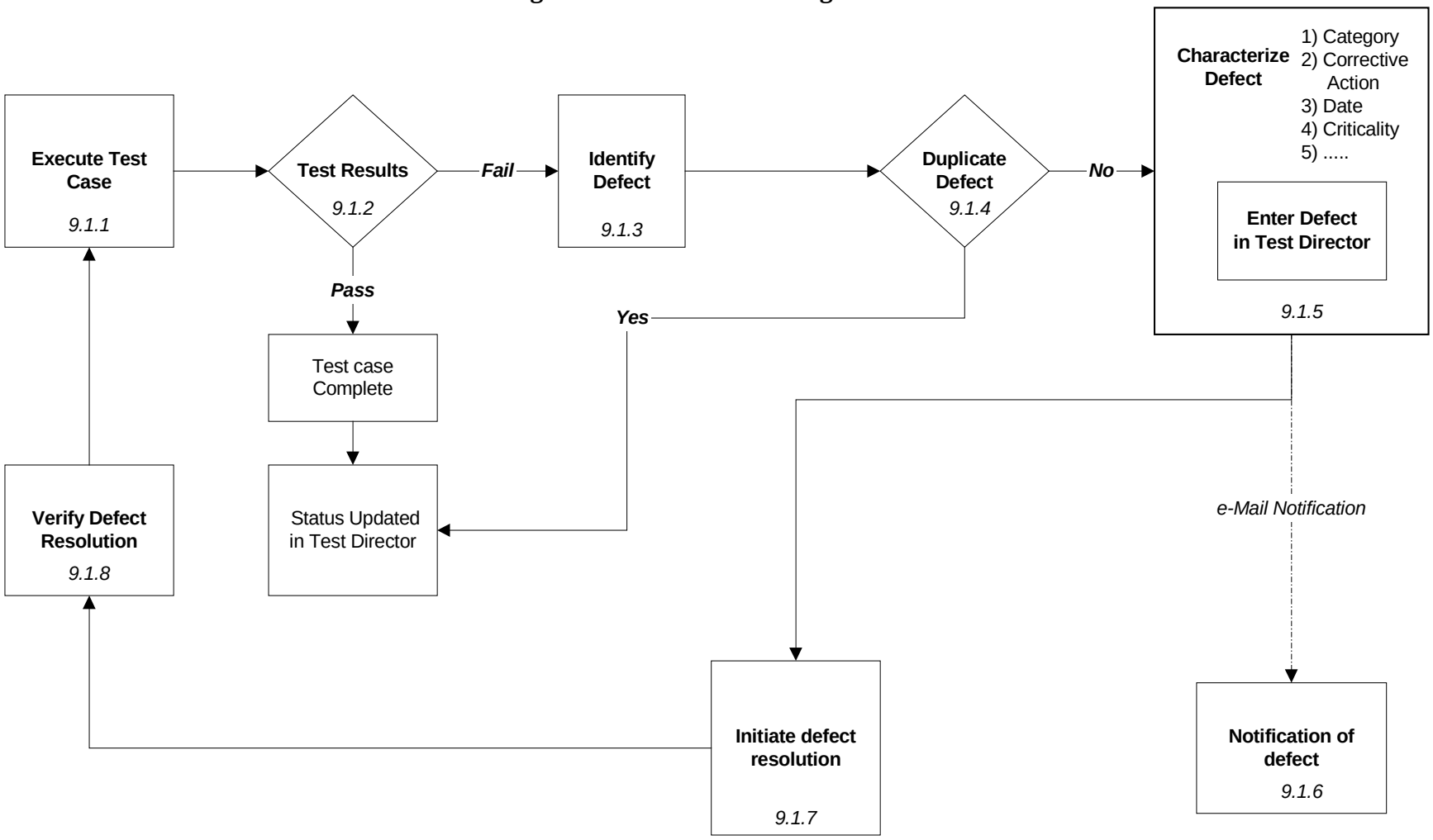
Defects will be tracked using TestDirector during the User Acceptance Testing effort. Refer to the following section entitled Defect Tracking in this document for further discussion of defects.

The exit and acceptance criteria for the User Acceptance Testing effort will be described in APP-076 User Acceptance Test Plan.

9.0 DEFECT MANAGEMENT

When actual test results do not match expected results, a defect is reported. Defects for the NEMAIS project will be recorded, tracked, cleared and reported using Mercury Interactive's TestDirector tool. The NEMAIS defect tracking workflow is illustrated in Figure 9.0-1 and described in the following subsections.

Figure 9.0-1 Defect Tracking Process



9.1 Defect Tracking Workflow

A number of steps are involved in identifying, reporting and resolving defects, each of which must be completed for every defect discovered during a test phase. The following sub-sections provide a summary description of each defect tracking workflow step.

9.1.1 Execute Test Case

The tester initiates action to create a test run to validate one or several test cases or test sets. A test run may be required due to a new transport, regression testing, creation of a new test set corresponding to a scenario or to verify defect resolution.

9.1.2 Test Results

The results of each test will be evaluated by the tester executing the test. The Pass/Fail criteria should be based on whether the actual results match the expected results. Once the status of the test is determined the tester should follow the applicable path as illustrated in Figure 9.0-1.

9.1.3 Identify Defect

Tests that result in a 'Fail' status should be evaluated to determine the cause of the failure. The tester will identify the cause of the failure and record a succinct summary of the defect. This will allow the tester to check for any duplicate or similar defects previously documented in TestDirector as discussed in the next section, 9.1.4.

9.1.4 Duplicate Defect

TestDirector provides the ability to search for similar or duplicate defects already documented in the defect tracking repository. The tester will use this unique functionality to locate similar or duplicate defects, previously documented in TestDirector, to ensure defects are only tracked once within the defect tracking system. If no duplicate or similar defect exists, the tester will proceed to the next section as illustrated in Figure 9.0-1. If a duplicate defect is located, the tester will not complete the entry of the defect into TestDirector and will inform the Test Leads of the duplication. The Test Leads will confirm that the defect is a duplicate and check the resolution status of the initial defect documentation and update the status of the defect in the TestDirector defect tracking system.

9.1.5 Characterize Defect and Enter into TestDirector

After the tester executes the test run and compares actual results to expected results, the tester will enter identified defects, including duplicates, in the automated test tool manager TestDirector. A number of system fields are configured in TestDirector to capture defect information and additional user fields will be created to capture project specific information. Prior to completing the entry of a defect in TestDirector, the tester should ensure that the defect has been characterized by entering the applicable data in these system and user fields at the time of entry. Details of the defect data elements and procedures to assign value to the data elements will be described in a later document or added as an Appendix to this document after the automated testing tool and requirements management tool have been installed and configured.

9.1.5.1 Categorize Defect

Defects will occur due to a number of different causes. It is incumbent upon each tester to ensure the root cause of every test failure is correctly identified and categorized. Correctly distinguishing between the types of defects and implementing the appropriate resolution is a key factor in ensuring the quality of the solution is maintained. Defects for this project will be categorized as listed in Table 9.1.5.1-1. Included with each category of defect are suggested resolution approaches.

Table 9.1.5.1-1 Defect Category and Suggested Resolution Approach

Defect Category	Criteria and Suggested Resolution Approach
Design Defect	The system actually meets the functionality documented and managed in the requirements traceability tool (Telelogic's DOORS); however, the documented requirements are incorrect due to an omission or misinterpretation of the user requirements. Resolution - Identify the defect, evaluate the defect to determine the correct set of requirements and appropriate design change, manage the defect for correction in the current or future project phase, and retest.
Configuration Defect	The configuration made as part of the system solution is defective in that it does not meet the user functional requirements documented and managed in the requirements traceability tool (Telelogic's DOORS) or it fails to function. Resolution - Identify the defect and assign the responsibility for resolution to the applicable Module Team. The Module Team will resolve the defect during Phase A and the tester will retest the corrected configuration. The defect will be cleared after validating the test.
ABAP Coding Defect	The ABAP program is defective in that it does not meet the user functional requirements documented and managed in the requirements traceability tool (Telelogic's DOORS) or the program fails to function. Resolution - Identify the defect and assign the responsibility for resolution to the Programming Team. The Programming Team will resolve the defect during Phase A and the tester will retest the corrected program. The defect will be cleared after validating the test.
Vendor Software Defect	The COTS software does not operate as documented. Resolution - Identify the defect, assign the responsibility for notifying SAP to the applicable Module Team. The Module Team will be responsible to work with SAP to resolve the defect during Phase A. The tester will retest the corrected software at the first available opportunity after receiving notification from the Module Team that the defect is resolved. The defect will be cleared after validating the test.
Testing Defect	The tester executed the test case incorrectly, or misread/misinterpreted the results of the test, or the script/scenario was ill constructed. Resolution - Identify the defect, correct the defect and re-perform the test case.

Defect Category	Criteria and Suggested Resolution Approach
Data Quality Defect	There was a defect in the incoming event data resulting in the test anomaly. Resolution - Identify the defect, correct the faulty data and re-perform the test.

9.1.5.2 Assign Criticality Level

The following is a high level discussion on the process of assigning a level of criticality to a defect. The detailed process that will be used for the NEMAIS project will be initially identified in the Phase A Integration Test Plan and will be incorporated in all future NEMAIS project test plans. The assignment of criticality levels to each defect documented in TestDirector will be determined by the members of the Triage Team. The Triage Team is discussed in more detail in section 10.0 of this document.

Criticality level numbers range from 1 to 5, with criticality level 1 being the highest and 5 being the lowest. The Module Leads should use criticality level numbers for allocating resources to fix the problems found. The criticality levels to be used and their meanings are identified in Table 9.1.5.2-1. The criticality levels identified are based on documentation in the IEEE Standards Collection Software Engineering 1997 Edition.

Table 9.1.5.2-1 Defect Criticality Levels

CRITICALITY	APPLIES IF A PROBLEM COULD:
1	**CRASH/LOCK-UP** a. Prevent the accomplishment of an operational or mission essential capability b. Jeopardize safety, security, or other requirement designated "critical"
2	**TECHNICAL PROBLEM (major) with No WORK AROUND** a. Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known
3	**TECHNICAL PROBLEM (major) with WORK AROUND** a. Adversely affect the accomplishment of an operational or mission essential capability but a work-around solution is known b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known

CRITICALITY	APPLIES IF A PROBLEM COULD:
4	**TECHNICAL PROBLEM (minor)** a. Result in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capability b. Result in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of those responsibilities
5	**MINOR -** Any other effect, cosmetic problems

9.1.6 Notification of Defect

As previously discussed in section 5.1, TestDirector will be configured to automatically notify designated users via a rules based e-mail notification feature that will be managed by the Test Leads. TestDirector also provides the ability to manually e-mail defect information to selected users, which will be implemented as required by the Test Leads.

9.1.7 Initiate Defect Resolution

Defect correction for software defects will be the responsibility of the Module and/or Programming Team, as applicable. Defect correction for other types of defects listed may require Module, Programming, Data Migration and Testing Teams to correct the defect. Correction of defects should occur as prioritized by the Triage Team and discussed in section 10.0.

The applicable team leads will be responsible for timely correction of defects identified during each phase of testing to ensure that the project, and particularly the testing process, stay on schedule.

When a defect is taking longer to fix than expected, depending upon the nature of the defect, a procedure for escalation has been defined in section 10.0 (Triage Team/Escalation Procedures) to address timely defect correction. In addition, any defects/issues that cannot be resolved will also require escalation.

Additional details identifying the process to assign a criticality level to a defect and the responsibility for assignment of criticality levels will be addressed in the test plans for Integration Test, Systems Test and User Acceptance Test.

All critical defects will be fixed or resolved prior to going live and will be addressed for resolution according to the criticality assigned by the Triage Team.

9.1.8 Verify Defect Resolution

After the defect is identified as resolved, the Testing Team will verify that defect corrections have been successfully completed. The resolution will be verified by adding the associated test to a new test case as illustrated in Figure 9.0-1. The corrective action will be deemed successful and assigned a status of ‘Pass’ only when the actual results match the expected results. In all cases, the Testing Team member performing the test will notify the Test Leads of the Pass/Fail status of the defect correction testing and the Test Leads will update the status of the defect in

TestDirector. Only the Test Leads will have the authority for closing defects.

9.2 Defect Reporting

The defect tracking database maintained in Test Director will be used to generate reports that can be reviewed by project management, the project team and stakeholders. These reviews may be performed to evaluate the progress of the project, the quality of the solution, and the likelihood of the solution meeting quality targets before implementation. Table 9.2-1 lists a sampling of the reports available from Test Director for defect analysis. Other reports will be customized in TestDirector to generate reporting information required for this project.

Table 9.2-1 Defect Reports from TestDirector

View	Description/Purpose
Open Defects by: • Priority • Severity • Assigned to (Team) • Assigned to (Individual) • Detected By	• Shows all defects by the views listed • Used for periodic reporting when a full view of open defects is required.
Pending/Re-Test by: • Priority • Severity • Assigned To (Team) • Assigned To (Individual) • Detected By	• Shows all defects migrated to system test by the views listed
Closed Defects by: • Priority • Severity • Assigned To (Team) • Resolved By • Re-Tested By • Reason Code	• Shows all closed defects by the views listed • Used when a full view of closed defects is required
Late Defects by: • Priority • Severity • Assigned To (Team) • Assigned To (Individual)	• Shows all late defects by the views listed • Used for program level reporting
All Defects by: • Priority • Severity • Status • Assigned To (Team) • Assigned To (Individual)	• Shows all defects by the views listed • Used to view defects regardless of status

10.0 TRIAGE TEAM/ESCALATION PROCEDURE

A Triage Team will be established to assign defect criticality levels and determine the prioritization and resolution of defects and of testing related issues that arise during the Integration Testing and User Acceptance Testing efforts. The team will meet daily during the test effort and will be chaired by the Test Leads. Issues may include such items as allocation of resources, schedule requirements, and any factor that may impact the satisfactory completion of the testing effort. The team will be comprised of representation from all Modules, Security, Nuclear, IT, ABAP and Basis, Change Management, Project Management and Testing. Individuals assigned to the Triage Team must be technically and functionally knowledgeable of the solution processes to ensure an accurate resolution of each issue. Details governing the specific responsibilities of this team during each testing effort will be amplified in the respective test plans.

The testing effort and resolution of defects and testing related issues should occur during the working core hours as defined in POPP 008 Core Hours. Resources should be available (on-call) after core hours to address defect resolution. A point of contact (POC) should be assigned to coordinate defect resolution within each team and should have the responsibility of contacting on-call personnel to support resolution of defects after core hours. The use of the on-call process will be discretionary and will be based on the priority/impact level assigned each defect.

All test defects and testing related issues identified during Integration Testing and User Acceptance Testing should be resolved as discussed in the following paragraphs

Since NEMAIS is a multi-phased project that may allow or require that certain defects be deferred for resolution in a later phase, the Triage Team must identify candidate defects for deferral and process them for approval. Defects that do not qualify as candidates for deferral should be resolved as discussed in section 9.1.7.

Defects proposed for resolution in a later phase must be approved through the escalation process managed by the Triage Team. Defects approved for deferral to a later phase will be annotated as such by the Test Leads in TestDirector. Defects proposed for deferral, but not approved, should be resolved as discussed in section 9.1.7.

The primary function of the escalation process is to alert senior management to a defect/issue requiring resolution that may adversely impact testing and the project schedule. The process should also be used when a recommendation to postpone the resolution of defects/issues is submitted. The Triage Team will assign a priority level to each defect/issue at the first Triage meeting following discovery of the item. The priority level assigned should be based on the impact the item has on the project delivery schedule and the criticality level assigned in section 9.1.5. The priority/impact level assigned should assist senior management in making informed decisions that will facilitate the effective and efficient resolution of the defect/issue.

Escalation of defects/issues should be based on the priority/impact level assigned by the Triage Team. The following addresses the expected steps and associated time frame for resolving/escalating a defect/issue.

High Priority/Impact level defects/issues should be resolved as discussed in section 9.1.7 within one working day following the Triage meeting that assigned the priority/impact level. Preliminary assignment of a High Priority/Impact level to a defect/issue prior to the Triage meeting is considered appropriate and will be reviewed by the Triage Team to confirm the assignment. Team Leads are expected to take action to facilitate the immediate resolution of the item. The escalation process will be invoked if the item is not resolved at the end of the first day and will be escalated to the next level for each day the item remains unresolved.

Medium Priority/Impact level defects/issues should be resolved as discussed in section 9.1.7 within two working days following the Triage meeting that assigned the priority/impact level. The escalation process will be invoked if the item is not resolved at the end of the two days and will be escalated to the next level for each day the item remains unresolved.

Low Priority/Impact level defects/issues should be resolved as discussed in section 9.1.7 within three working days following the Triage meeting that assigned the priority/impact level. The escalation process will be invoked if the item is not resolved at the end of the three days and will be escalated to the next level for every two days the item remains unresolved.

The escalation timetable, based on defect/issue priority/impact, is summarized in Table 10.0-1. Depending on schedule progress, the time frames in Table 10.0-1 may be compressed as determined by the Triage Team.

Table 10.0-1 Escalation TimeTable

Priority /Impact	Expected Resolution Time	Level 1 Escalation Team Leads	Level 2 Escalation Domain Manager	Level 3 Escalation Project Executives
High	1 Day*	+1 Day*	+1 Day*	+1 Day*
Medium	2 Days*	+1 Day*	+1 Day*	+1 Day*
Low	3 Days*	+2 Days*	+2 Days*	+2 Days*

* 1 Day = core hours workday as defined in POPP 008.

11.0 TESTING METRICS

As stated in the response to RFQ N00024-00-Q-5229, status metrics fall into one of two broad categories:

- Measures of effectiveness that gauge the solution's success, and
- Measures gauging the health of the NEMAIS development effort

It is the intent of the Testing Team to assist the NETS in meeting the objectives of providing the best value to the Navy and maximizing customer satisfaction by implementing a functional metrics process for testing. The Testing Team will use and capture metrics during each testing

phase to facilitate the management of the "go-live" strategy and to measure the quality and efficiency of the testing process. Mercury Interactive's TestDirector and Telelogic's DOORS will be used to track and report on the metrics captured for the NEMAIS ERP solution. Several of the key metrics that can be captured and managed through TestDirector and Telelogic's DOORS include but are not limited to those listed in Table 11.0-1.

Table 11.0-1 Suggested Metrics

Report	Description
Requirements Traceability Matrix	Provide data to identify whether enough of the application has been tested
Test Case Creation Completeness	Provide information that allows monitoring of the status of test preparedness.
Defect Status Report	Provides information to support monitoring the status of each documented defect with respect to new/open/assigned/fixed/re-tested/closed deferred.
Defect Arrival/Closure Rate	Provide the data to help analyze the trend of defect detection and closure to allow for an informed decision on the risk of releasing the solution.
Defect Removal Efficiency	Provides the data to support monitoring the rate of defect correction, which can assist in identifying work assignment disparities or staff burdens as well as pinpoint potential design or requirement deficiencies.

After a complete analysis of the functionality of TestDirector and Telelogic's DOORS, in conjunction with the release of the NETS project Implementation Strategy, a more detailed list of metrics will be documented in the test plans created for Integration, User Acceptance and System Testing as they apply to the project and each test phase.

12.0 TRAINING

Fifteen (15) days of on-site consulting for the Mercury Interactive tool suite have been procured for this project. A portion of this consulting time will be used for training. The specific training plan and content of training will be developed by the Change Management Team and Testing Team in conjunction with Mercury Interactive after the tool set has been installed and after the Testing Team, in conjunction with the IT Infrastructure Team, has ensured that the tool is appropriately configured. This training will occur before Module Teams are required to begin recording unit tests.

13.0 ROLES AND RESPONSIBILITIES

The roles and responsibilities of NETS team members regarding test efforts have been addressed throughout the document. For clarification, Table 13.0-1 summarizes these roles and responsibilities.

Table 13.0-1 Roles and Responsibilities

Role	Responsibilities
Test Team	1) Develop automated testing scenarios for regression testing of business processes in the TST instance 2) Manage the execution of automated tests unattended 24 hours a day, if required 3) Manage defect tracking and reporting in TestDirector 4) Generate test related metrics from TestDirector 5) Manage the links between TestDirector and Telelogic's DOORS 6) Manage the automated and manual defect notification process in TestDirector 7) Manage the LoadRunner test tool to support System/Stress testing 8) Manage the clients in the TST instance 9) Create and manage user fields in TestDirector 10) Implement configuration management for unit tests after the associated configuration is transported to the TST instance 11) Re-execute the unit test after the satisfactory transport of the associated configuration to the TST instance. 12) Manage pre-integration defects 13) Parameterize unit tests in preparation for integration testing 14) Verify operability of parameterized unit tests 15) Build scenarios by grouping parameterized unit tests 16) String the business scenarios to perform end-to-end integration testing. 17) Develop Integration Test Plan and User Acceptance Test Plan 18) Coordinate the planning and performance of System/Stress testing 19) Participate in the selection of tests to be used for UAT 20) Track and report UAT defects encountered 21) Manage and conduct the UAT effort 22) Identify exit and acceptance criteria for integration and user acceptance testing. 23) Have sole responsibility for annotating defects as resolved in TestDirector. 24) Make determination of when to stop or continue testing based on defect quantity and criticality 25) Chair the Triage Team

Role	Responsibilities
Module Team	1) Record unit tests using either WinRunner QuickTest for R/3 or WinRunner in the DEV instance 2) Save recorded tests in TestDirector to support test management and the automatic creation of design steps 3) Save manually created tests in TestDirector 4) Enter descriptive test information in TestDirector 5) Review and modifying design steps as required for completeness and accuracy 6) Perform Unit Configuration Testing 7) Verify each unit test prior to keying the associated configuration into Client 005 of the DEV instance. 8) Verify each unit test prior to the transport of the associated configuration to the TST instance. 9) Resolve defects 10) Provide the information required to allow the grouping of tests to create business scenarios. 11) Assist in development of the Integration Test Plan 12) Provide personnel to support the integration test effort. 13) Provide representation on triage team
Integration Test Team	1) Perform Integration Testing 2) Identify, document and verify uniqueness of defects 3) Capture and document testing data metric
Project Team	1) Provide representation on triage team 2) Support the escalation process
Change Management Team	1) Identify the individuals to perform the User Acceptance Testing 2) Schedule the UAT participants availability to perform the testing 3) Train the UAT participants 4) Provide input into the selection of tests to be used for UAT 5) Create documentation to be used for training and execution of User Acceptance tests 6) Support conducting the UAT effort 7) Define UAT feedback, other than defects, through the Regional Representatives 8) Provide representation on triage team
Programming Team	1) Record unit tests using either WinRunner QuickTest for R/3 or WinRunner 2) Save recorded tests in TestDirector to support test management and the automatic creation of design steps 3) Enter descriptive test information in TestDirector 4) Review and modifying design steps for clarity

Role	Responsibilities
	5) Perform Unit Testing of Development 6) Provide representation on triage team
Basis Team	1) Assist in the planning and performance of System/Stress testing 2) Perform disaster recovery testing 3) Perform backup and recovery testing 4) Perform failover testing 5) Perform Hardware/Software validation testing 6) Provide representation on triage team
Infrastructure Team	1) Provide resources to support establishment of a test facility prior to commencing integration testing. 2) Assist in the planning and performance of System/Stress testing 3) Perform disaster recovery testing 4) Perform backup and recovery testing 5) Perform failover testing 6) Perform Hardware/Software validation testing 7) Provide representation on triage team 8) Develop Infrastructure test plan and perform Infrastructure testing
Triage Team	1) Assign defect criticality levels 2) Determine prioritization of defects and testing issues 3) Determine resolution of defects and testing issues 4) Manage the escalation process 5) Compress escalation process time frames as dictated by schedule requirements
Requirements Management Lead	1) Manage requirements and business processes in Telelogic's DOORS

A. Acronyms and Abbreviations

ABAP	Advanced Business Application Programming
ABC	Activity Based Costing
AUT	Application Under Test
CATS	Cross Application Time Sheet – part of HR SAP module
CM	Configuration Management
CO	Controlling SAP Module
D level	Depot Level Maintenance
DEV	Development instance of SAP
DM	Document Management
DOD	Department of Defense
EHS	Environmental Health and Safety
FI	Finance SAP Module
FM	Funds Management SAP Module
GUI	Graphical User Interface
HR	Human Resources SAP Module
I Level	Intermediate Level Maintenance
IPT	Integrated Product/Process Team
IS-PS-2	Industry Solutions – Public Sector –Federal Government Accounting SAP Module
IV&V	Independent Verification and Validation
MM	Material Management SAP Module
MS	Microsoft
NEMAIS	Navy Enterprise Maintenance Automated Information System
NETS	Navy Enterprise Team Ships
O Level	Organizational Level Maintenance
PM	Plant Maintenance SAP Module
PRD	Production instance of SAP
POPP	Program Office Policy and Procedure
PS	Project Systems SAP Module
QM	Quality Management SAP Module
QSS	Quality Systems and Software (software vendor)
RRC	Regional Repair Center
RT	Regression Test
RTM	Requirements Traceability Matrix
SAP	Systems, Applications & Products in Data Processing
SHO	Road Show instance of SAP
SSL	System Supplying the Load
SUT	System Under Test
TMS	Transport Management System
TRN	Training instance of SAP
TST	Integration Testing instance of SAP
UAT	User Acceptance Test
WF	Work Flow SAP Module

B. Definitions

Backup And Recovery Testing	Testing performed by the Infrastructure and Basis Teams to ensure that a system can be restored from a backup.
Configuration	In SAP configuration is the customization of system settings made to satisfy project governing design specifications.
Defect	When tests are created, expected results are documented. When tests are run and actual results do not match expected results, a defect is reported.
Disaster Recovery Testing	Disaster Recovery Testing will be planned and performed by the Infrastructure and Basis Teams to simulate a system failure during critical system processes and to demonstrate recovery of databases to a usable state.
Failover Testing	Testing performed by the Infrastructure and Basis Teams to ensure that in the event of a Production (PRD) instance failure the system will revert to and begin operating on the system designated for this purpose, which, for this implementation, is the Testing (TST) instance. Failover testing will affect any testing effort in progress on the TST instance and should be scheduled to minimize the impact on any testing performed on the TST instance.
Failure Component Testing	Testing performed by the Infrastructure Team to determine if all components of a given system are operational before the system is opened to users for use.
Hardware/Software Validation Testing	Testing performed by the Infrastructure and Basis Teams to ensure that the installed hardware, operating system software, drivers, patches and all server connectivity are fully functional prior to installation of solution software. This would also include Backup and Recovery Testing and Failure Component Testing.
Infrastructure Testing	Testing performed by the Infrastructure and Basis Teams which includes Failover Component, Hardware/Software Validation, Disaster Recovery, Failure Component and Backup and Recovery Testing. Documentation describing the details of Infrastructure Testing will be in a work product prepared by the Infrastructure Team.
Integration Testing	Testing performed by a team comprised of both the implementation partner and the Navy representatives to validate that all the software components of the implementation function properly together and

	operate according to the "to be" business process definitions and governing design specifications. In the Integration Test, a complete system will be built and tested from components and subsystems, including internal and external interfaces, and tested as a complete entity
Regression Testing	Selective re-testing of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with governing design specifications.
Scenario	A scenario is the equivalent of a business sub-process in Project Office. A scenario may consist of text descriptions as well as flowcharts used to graphically represent movement within the sub-process
Script	Scripts generally serve to document a work step or transaction. A script contains detailed instructions to execute a work step. Typically there are one to several scripts for a given work step. Scripts are used for testing and training. Occasionally the term script will be used for both the work step and the business sub-process.
Systems/Stress Test.	Testing performed by a combination of IT Infrastructure, Basis and Testing Teams to determine if the hardware components meet volume load requirements (load testing) and required performance levels (performance testing) across the entire architecture
Testing	Testing is the process of evaluating the proposed solution under specified conditions to detect the differences between existing and required conditions and documenting the results.
Transaction	A transaction is a work step that is executed on a system.
Transport	Transport is a SAP term: It is the definition of a movement of entities from one SAP instance or client to another.
Unit Configuration Testing	Testing performed by the Module Teams to determine if the configuration corresponds to the business process definitions and business requirements.
Unit Testing of Development	Testing performed by the programming staff to determine if the various function modules comprising a program operate according to the governing design specifications.
Unit Testing of Security Access	Testing performed by the Module Team to determine if the security access profiles correspond to the governing design specifications.

Profiles

User Acceptance Testing	Testing performed by stakeholder representatives to ensure that the implementation meets the user requirements. This testing will include functional as well as security testing. Once the tests are completed to an acceptable level the stakeholder representatives will approve the User Acceptance Testing as one of several steps required to approve the system for go-live. The User Acceptance Test (UAT) is used to demonstrate to and gain the confidence of the user community that the system meets their requirements and the project is ready for deployment. User Acceptance Testing will be performed on the Testing (TST) instance, which will mirror the intended production environment to assure that the testing results will reflect those in the live environment. Prior to commencing UAT, the stakeholder end user testers will be trained on the functionality of the system as well as the use of the Operating Procedures created to document the configuration designed for the "to-be" business processes.
Work step	Work steps further breakdown a business sub-process or scenario. A work step may be described by text, menu paths, and screen shots. Typically a work step is equivalent to a SAP R/3 transaction.